1981-01-27

PRELIMINARY SPECIFICATIONS FOR A DISTRIBUTED PACKET RADIO NETWORK FOR COMPUTER- AND RADIO-AMATEURS.

Jens Zander
Robert Forchheimer

INTERNAL REPORT
LITH-ISY-I-0424

Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden

Linköpings tekniska högskola
Institutionen för Systemteknik
581 83 Linköping

1981-01-27

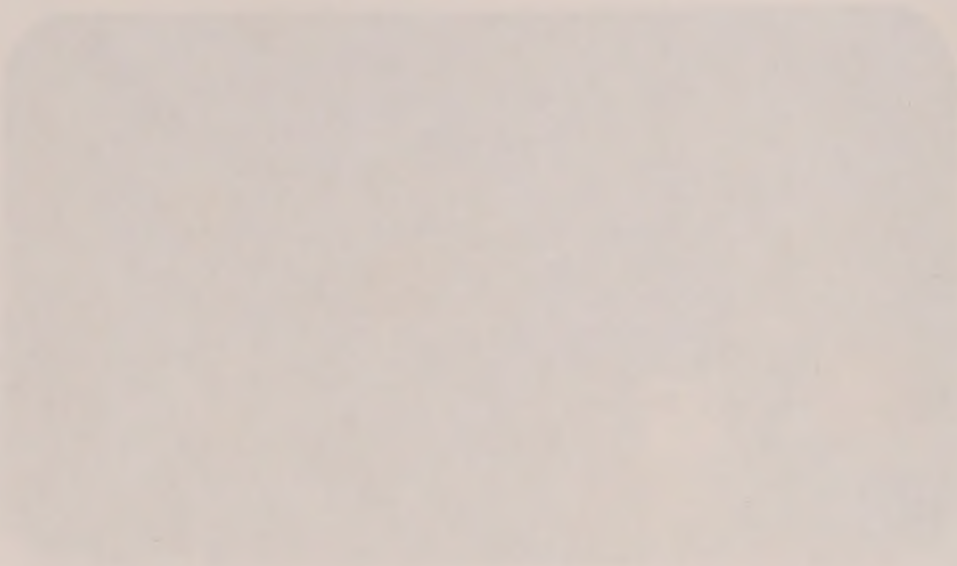
PRELIMINARY SPECIFICATIONS FOR A DISTRIBUTED PACKET RADIO NETWORK FOR COMPUTER- AND RADIO-AMATEURS.

Jens Zander
Robert Forchheimer

INTERNAL REPORT
LITH-ISY-I-0424

Abstract

This report presents preliminary specifications for an UHF-radio based packet switched computer network . The users of this network are radio- and computer amateurs with no commercial aims. Radio- (link-) level control, as well as routing and network control issues are considered, and sketches for possible solutions are presented. The concept of a control language for user network control is introduced. Finally, a flexible node-hardware layout is proposed, which meets the requirements of simplicity and low cost.



CONTENTS

PART I
INTRODUCTION

1. Introduction

2. General Network outline

2.1 Requirements
2.2 Network topology
2.3 Message switching

PART II
NETWORK MECHANISMS

3. Access Methods

3.1 General considerations
3.2 Analysis of access schemes
3.3 Suggested multiple access scheme

4. Routing

4.1 Soft structures
4.2 Initialization
4.3 Address and location specification

5. Network Control

5.1 Traditional control structure
5.2 Proposed control philosophy
5.3 A Control language
5.4 Implementation
5.5 Examples
5.6 Network integrity

Digitized by the Internet Archive
in 2025 with funding from
Amateur Radio Digital Communications, Grant 151

https://archive.org/details/prelimspecific

PART III
IMPLEMENTATION

6. Node Hardware

6.1 Control Unit
6.2 Packet Radio Transceiver

7. Preliminary NODE interface specifications

7.1 USER/NODE interface
7.2 LINK interface

8. FORTH Extensions

8.1 Packet handling
8.2 Protected dictionaries and vocabularies
8.3 External user vocabularies

9. Summary and Conclusions

10. References

APPENDIX
FORTH - an introduction

PART I

INTRODUCTION

CHAPTER 1

1. Introduction

Due to rapid growth in the field of Personal Computing there has been an increase in demand for better communications between computer hobbyists. The available federal and commercial telephone service represents a rigid and inflexible framework for the communication needs of the computer hobbyist. A possibility to solve these problems is to use the amateur-radio bands (or other suitable frequency bands) for a distributed packet radio network as outlined in [1] and further refined in this report. The network is considered to be used by radio-amateurs and computer hobbyists on a non-commercial basis. A network like this has to be designed for high flexibility to meet future demands.

This paper outlines an experimental radio network. The aim is to define the structure and suggest solutions to the vital problems. The proposed experimental network, built for and by hobbyists, can be used as a testbed for routing and access algorithms. Furthermore, such a network will provide the experimenter with a large number of active participants, which is essential for large scale experiments. In the following chapter we first present a general network outline and identify various types of problems. General solutions to these problems are discussed in Part II, and in Part III we touch upon implementational aspects.

CHAPTER 2

2. General network outline

2.1 Requirements

It is obvious that the requirements on a data network for the hobbyist and other non-commercial services are quite different from those on existing commercial networks. Service accounting and extreme reliability are heavily stressed in commercial networks. In our case, however, other aspects, particularly technical ones are of great interest. The users who communicate via the network are widely separated geographically and they are using personal computing equipment of moderate complexity. To the network they act as bursty data sources with long idle periods and moderate instantaneous input bit-rates (1-10kbit/s). Any user should be able to communicate with any other user, irrespectable of geographical distance, by means of this network. The network should be resistant to total failure, although performance reduction of short duration can be accepted. It should furthermore be flexible and allow new users to enter and others to exit without affecting the overall performance.

2.2 Radio network topology

The above requirements can be met by a radio based network as presented in [1]. In this network every user constitutes a node. Depending on their effective radio ranges, nodes communicate with a number of neighboring nodes. An example is shown in fig 1. Nodes which can communicate are connected. All paths are considered to be two-way. The effective range, and thus the number of neighbors that a node is communicating with, is a function of several factors. The topology of the surroundings of the node antenna (hills, lakes, cities etc.) plays an important role, as do the frequency band and the transmitter power used. The frequency bands available for these applications are located above 100 MHz which, when using moderate power (1-10W) and omni-directional antennas, will limit the range to 5 - 50 km.

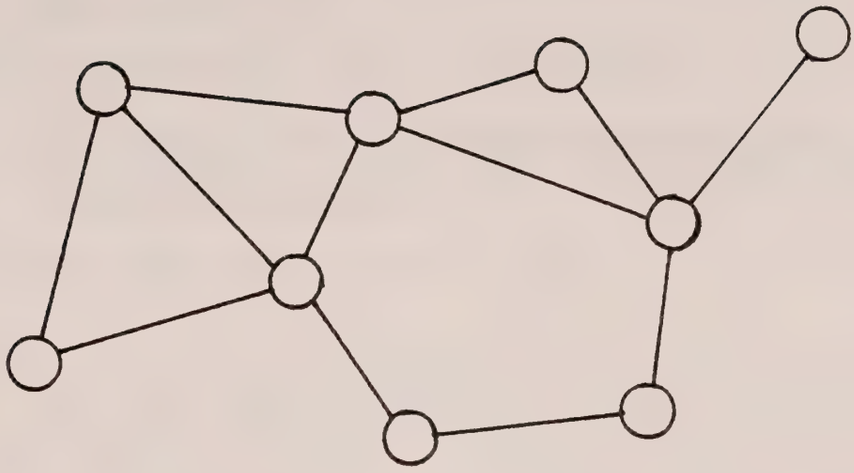


Fig. 1. Typical network graph

There is no natural hierarchy among the users so it is assumed that all nodes should be essentially alike. This implies that all message switching in the network has to be performed in a decentralized manner. Communication over greater distances than the range of a single node is achieved by using other nodes along the way as repeaters in a multi-hop fashion [3]. Furthermore, the example in fig. 1 should be considered as a snap-shot of the network at a single instant of time. New nodes enter the network and others leave, thus dynamically changing the network topology in a random way.

2.3 Message switching

In voice-communication different users are usually assigned channels to be used during the whole session. This is of course impractical in our case since the users present the network with very bursty traffic. We have earlier concluded that it seems more practical to use a packet switching system [1]. At the user's node a message is broken down to packets of moderate length (approx. 1000 bits) and information about the destination and origin of the message is attached. The packet is then presented to the network and by means of distributed algorithms forwarded to its destination. This approach makes it possible to break down the design problem into a few major areas :

1. Local communication (Channel access methods):

   how can neighboring nodes communicate without disturbing each other?

2. Global communication (Routing, flow-control):

   how should a packet be routed through the network in order to experience minimum delay and at the same time avoid congestion situations.

These problems are dealt with in chapters 3 and 4.

To be able to use simple, standardized node equipment throughout the network, as well as utilizing the available frequency spectrum optimally, we wish to use a single, high speed simplex channel for all transmissions. This has great impact on problem 1; neighboring nodes cannot always use the channel at will , particularly when the channel is busy. The effective range of the nodes is relatively small, however, so the channel can be reused some distance away without disturbances. Finally we have a third problem of great significance, namely that of

3. Network control:

   how do we control the network, change routing algorithms etc.? How do different users control the network to handle their messages?

These problems are considered in chapter 5. In Part III, finally, we take a look at the hardware and software complexity of a node and suggest some specifications.

PART II

NETWORK MECHANISMS

CHAPTER 3

3. Access methods

3.1 General considerations

The first problem to be solved, is how to enable the various nodes to communicate independently with their neighbors, while sharing the same radio channel. A class of access schemes which handle this situation are the RTDMA (Random Time Division Multiple Access) schemes. Most of these schemes exploit the fact that data traffic is bursty, i.e. that the peak-to-average bit rate ratio is very high (in the order of 100-2000). Beside the time division effect, the channel can be simultaneously used at several locations, provided their geographical distance is great enough. This is due to the limited range of the nodes.

Several RTDMA schemes for wireless communications have been proposed in recent years. The first in this field was suggested in 1970 by Abramson et al [2]. This was the so called ALOHA random access scheme. According to this scheme different nodes transmit their packets at demand (i.e. as soon as there is one to be transmitted). There is, of course, the probability that another node, within range of the receiving node transmits a packet at the same time. The situation that arises is called a conflict or collision. It is assumed that the nodes are able to detect this situation and, by using random delays retransmit their packets until they are successful. If the traffic intensity is low, collisions are rare and conflicts are handled rapidly. If, however, the traffic intensity increases, the number of collisions will rise and retransmitting nodes will tend to cause new collisions, worsening the situation. An overload situation arises (channel saturation).

To achieve higher throughput the trend has been to reduce and/or control the number of collisions. Example of schemes of this kind are the various CSMA (Carrier Sense Multiple Access) schemes [4]. Here the nodes, as the name implies, sense if the channel is busy before attempting to transmit a packet. This reduces the probability of conflict. A conflict can only occur if the channel is sensed free by a node during another nodes T/R (Transmit/ Receive) switch time (and/or propagation delay). Provided that all nodes are in range of each other, a CSMA scheme can give a channel utilization of more than 80%, provided that switch-delays can be kept low. This can be compared with a maximum of 38% for the ALOHA type schemes.

In a network of the proposed kind the maximum throughput in a neighbor-to-neighbor path will be somewhat lower. This is due to the fact that not all nodes are in range of each other. Consider the case shown in fig. 2.

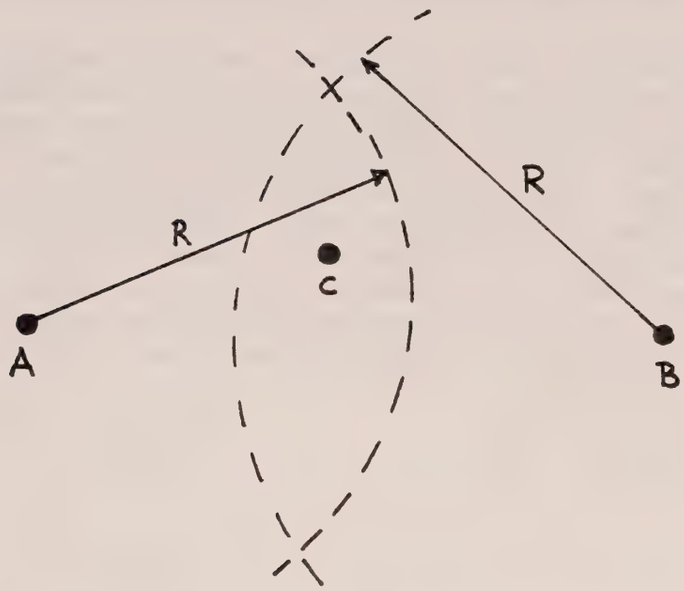


Fig. 2. Hidden terminal problem

The nodes A and B simultaneously wish to transmit a packet to node C which is within range (R) of both of them. A and B however are not able to detect the transmission of the other. The risk for a collision is obvious. Suppose A transmits his packet. B is not able to detect that the channel is in use and will also start to transmit his packet, causing a collision at C. This problem is usually referred to as the hidden terminal problem. In the following we will suggest a conflict reducing RTDMA scheme which recognizes this problem.

3.2 Analysis of access schemes

In order to design and evaluate an efficient access scheme for the network, some model assumptions have to be made. In the literature several schemes for packet radio communication have been analyzed. To adopt these results to our case, we need to make the problem somewhat more precise by introducing some notation and a simple model of the network situation.

It seems reasonable to use a planar point process to model the geographical distribution of the users (nodes). For simplicity we assume that this point process is a planar Poisson process with density parameter γ (nodes/km^2). Further, we approximate the total traffic emanating from each node with a Poisson process with parameter s' (packets/sec). Assuming that in a steady state condition each node receives the same number of packets as it transmits, we are able to determine how much of this traffic originates from the node itself and how much is repeated traffic. If we, denote by P_r, the probability that an incoming packet is retransmitted, and by λ the nodes' own traffic intensity, we can equate :

$$s' = P_r\, s' + \lambda$$

$$s' = \frac{\lambda}{1 - P_r} \qquad (1)$$

We denote all the traffic intensity in the surroundings of some receiving node by s. If we use an access scheme that results in conflicts, packets have to be retransmitted. It can be shown [4], that the resulting process is approximately a Poisson process with parameter g. We call this process the total offered traffic . We normalize all these parameters with respect to the packet delivery time (T), and use the capitals Λ,S,G to denote normalized values. The probability P_r depends on the number of hops a packet experiences on its way to its final destination through the network. In other words we model the number of repetitions as a geometrically distributed stochastic variable. The actual value of P_r is mainly a function of the habits and demands of the users. Do they tend to make "local" or "long distance" calls? The final answer to these questions can only be given by the experimental network itself. In the following we suggest and analyze a simple CSMA scheme, suitable for our network.

3.3 Suggested Multiple Access Schemes

The access algorithm is summarized by the flowchart in fig 3. Suppose a node has a packet ready for transmission to a neighboring node. The transmitting node starts by sensing the signal (carrier) level on the channel. If the channel is sensed free, it waits for td seconds. td is a random variable of some suitable distribution, but is always greater than t_a, the T/R-switch time. If the channel is again sensed free, the packet is transmitted and the node waits for an acknowledge packet.

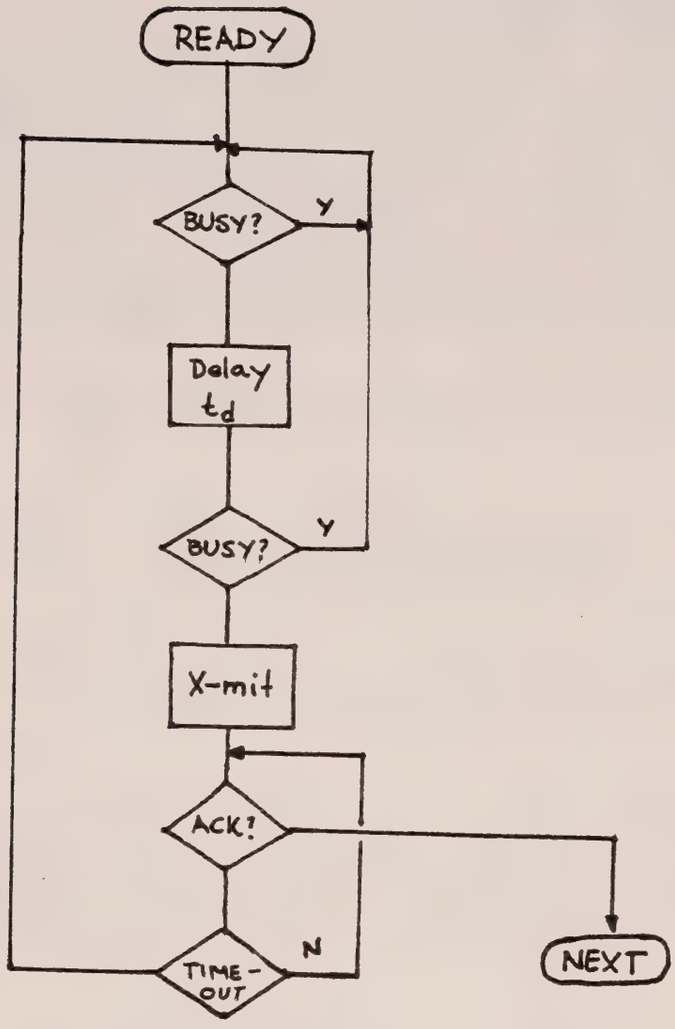


Fig. 3. Simple CSMA access algorithm

If the channel is busy at any of the sensing instants, the procedure is restarted from the beginning. Thus the node waits until the channel becomes free, or a packet is detected which is destinated to the node itself, in which case it enters the corresponding receiving algorithm. When the channel eventually becomes free, the node repeats the sensing and delay procedure in order to transmit its packet. After transmission it waits for an acknowledgement from the receiving node. If it is received, the packet is dropped and the node proceeds with the next packet in the transmitter buffer. If no acknowledgment or a NACK arrives it is assumed that a collision occurred on the channel and the procedure is started from the beginning in order to retransmit the packet. This is repeated until an ACK is received or the maximum number of allowed attempts is exceeded. It

should be observed that the delay t_d ($>t_a$) makes sure that acknowledge transmissions are not interferred.

The performance of a scheme like the one described can be evaluated fairly easily [4] provided there are no hidden terminals . The system proposed here will have slightly worse throughput/delay performance than a system where all nodes are in range of each other. Thus a system using the corresponding non-persistent CSMA scheme with no hidden terminals will provide an upper bound on the performance of our system. On the other hand, to derive a lower bound on performance, we can study a system using a unslotted ALOHA-scheme. This scheme does not utilize carrier sensing at all and is thus indifferent to the existence of hidden terminals. However, every sensible form of carrier sensing will reduce the number of collisions, and thus improve performance. We can therefore expect that the proposed system will outperform the ALOHA-type system even in the presence of hidden terminals.

To calculate the performance bounds, we introduce the following notation

T_p Packet duration
T_r Acknowledge packet duration
a T/R switch + propagation delay time

$T_p + T_r + a = 1$,
i.e. all durations are normalized with respect to T, the packet delivery time.

In the case of no hidden terminals CSMA (the upper bound) we can directly derive its throughput characteristics. Since, in this case no nodes are allowed to interfere with the acknowledge transmission, we consider both the actual transmitted packet and the acknowledgment as a single message of duration T. We get the throughput (1) S_u as [4]:

$$S_u = \frac{G \exp(-aG)}{G(1+2a) + \exp(-aG)} \qquad (2)$$

Assuming $a<<1$ we get the lower bound S_l from the ALOHA scheme assumption [4]:

$$S_l = G \exp(-2G) \qquad (3)$$

To evaluate (2) and (3) we have to evaluate the parameter a. The expected hop distances are in the order of 5 - 50 km, corresponding to propagation delays of 15 - 150 us. The T/R switch delay is in the order of 1 ms, and thus dominates a. Assuming a channel bit-rate of 100 kbit/s and data packet lengths of 150 bytes and 30 bytes for acknowledge packets, fig. 4 shows the approximate upper and lower bounds for the channel throughput for T/R-switch delays 1, 2, 5 ms. These T/R delays correspond to a= 0.06, 0.12 and 0.30. Note that the lower bound does not depend on the T/R switch time. Fig. 5 shows the corresponding packet delay bounds.

---

(1) The throughput S is the number of correctly delivered packets per packet time.

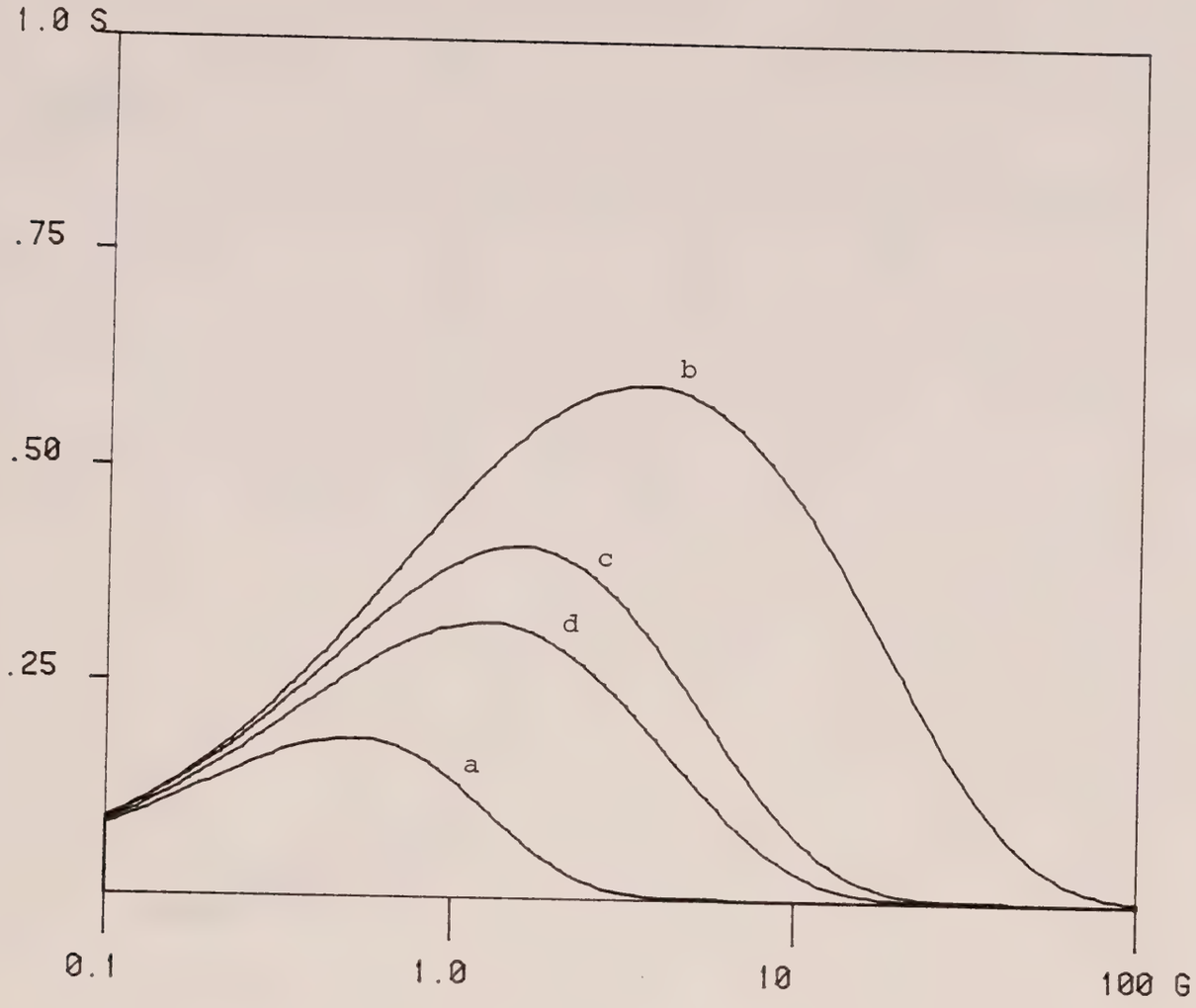


Fig. 4 Channel throughput bounds. S and G are normalized with respect to the message duration.
a) ALOHA system (lower bound)
b) np-CSMA ts=1 ms (a=0.06) (upper bound)
c) np-CSMA ts=2 ma (a=0.12) (upper bound)
d) np-CSMA ts=5 ms (a=0.30) (upper bound)

In figure 4 we can clearly see the potential instability of the system. For every channel throughput S, there are two values for the total offered traffic G, of which the lower one is desired (giving the shortest delay times, fig. 5). Only at the maximum point of the curves we have a potentially stable system. When operating at a relatively low input S at the low G equilibrium point, a sudden burst can make the system attain the high G operating point for the same input S. This is of course due to the

fact that retransmitted packets collide.

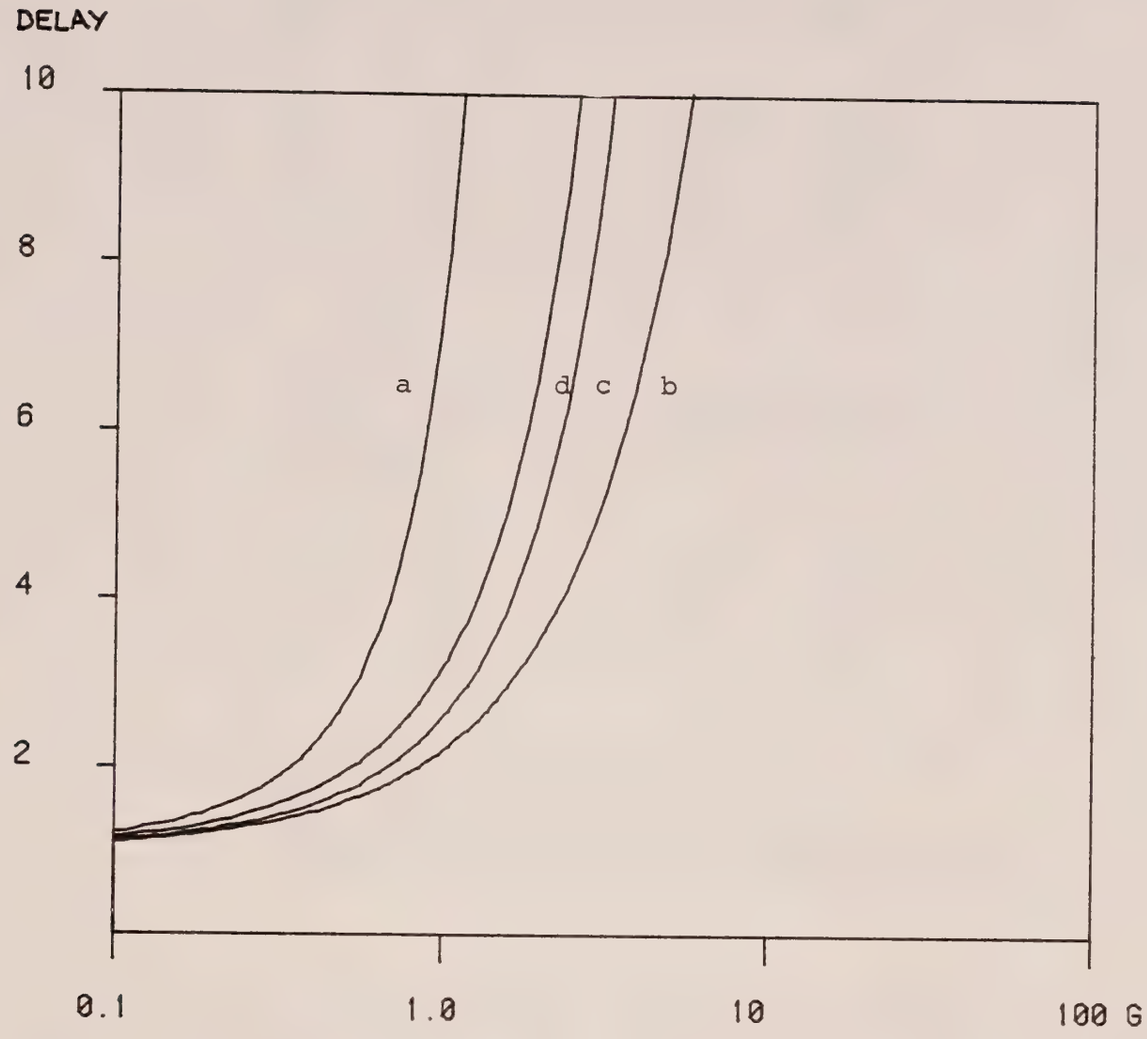


Fig. 5. Packet delay bounds
a) ALOHA system (lower bound)
b) np-CSMA ts=1 ms (a=0.06) (upper bound)
c) np-CSMA ts=2 ma (a=0.12) (upper bound)
d) np-CSMA ts=5 ms (a=0.30) (upper bound)

From the two figures, together with estimations of the user density (γ) and the user packet generation intensity (Λ), the throughput/delay characteristics of the single user can be evaluated. Table 1 gives some examples. N is the average number of users in the surrounding of a node ($N=\gamma\Pi R^2$, where R is the radio range of the nodes). The table shows the maximum average packet rate (Λ) and the corresponding bitrate for a channel

bitrate of 100 kbit/s and a packet message length (duration) of 1500 bits.

N=10

Pr	Packet rate (Λ) min (* 10E-2)	max	Bitrate (bit/s) min	max
.2	1.4	3.8	1300	3400
.4	1.1	2.8	950	2500
.6	0.7	1.9	630	1700
.8	0.4	1.0	310	840
.9	0.2	0.5	160	420

N=20

	Packet rate (Λ) min (* 10E-2)	max	Bitrate (bit/s) min	max
.2	0.7	1.9	630	1700
.4	0.6	1.4	470	1260
.6	0.4	1.0	320	840
.8	0.2	0.5	160	420
.9	0.1	0.2	80	210

Table 1. Maximum single user packet- and bitrates
for upper (max) and lower (bounds).
T/R delay = 2 ms, 1500 bit/message
Packet rate is normalized (packets/message-interval)

These results depend heavily on the initial assumption, that the sum of the nodes own packets, the repeated packets and all retransmitted packets on the channel form a poisson process. This will indeed hold true if the number of nodes N goes to infinity, retaining a constant total input to the channel. Furthermore, we only describe the steady-state conditions of the network.To be able to verify these results and to study transient and stability properties, a simulation model of the network has to be created. Since the proposed scheme is probably a conservative one, simulation results can also imply modifications and refinements. The possibility of using side information, like measuring the local traffic intensity, leads to adaptive multiple access schemes and should also be considered [5]. Such schemes can also be used to stabilize the system. In [1] an adaptive transmitter power controlling scheme is proposed.

CHAPTER 4

4. Routing

In [1, chapter 3] a number of low level routing algorithms are presented. They assume very limited packet and node routing information. They are robust to dynamic changes of the network and require very limited processing at each repeating node. The produced routes may however not be the shortest possible ones as the algorithms fail in taking into account non-euclidean "short-cuts" such as those offered by nodes linked by directional antenna paths or other long-distance media. Furthermore, in developing more sophisticated schemes we would like to utilize specific information about the network such as its main topology (e.g. the skeleton [1]) and state (congestion etc.) to improve routing decisions. Finally we would like to impose artificial structures upon the network such as the concept of stations [3] and point-to-point links (circuit-switching emulation). Generally these structures have no corresponding hardware components. In the following we will call them soft structures to emphasize that they are implemented by node programming.

4.1 Soft structures

A number of different philosophies exist with regard to the routing problem. Some of these assume that the network can be conditioned for a particular transmission situation. In these cases the nodes are temporarily programmed and required to keep their programs available for the duration of that specific situation. Such routing techniques are valuable when packets are to be transmitted along particular paths in a short time as in the case of a file transfer. To create the particular route one may use a path finding packet which travels according to the previously mentioned low level algorithms. Another alternative is to collect enough network information at the transmitting node to know in advance the whole path. Thirdly, we can use the concept of stations. These are nodes which possess more information than the average node about the network, either due to deliberate intent or to the fact that they are "old" and have accumulated a useful amount of network knowledge. The stations can be used as databases by other nodes and can supply routing information. Contrary to the classical concept, the stations do not necessarily handle the transmitted packets themselves. However they can be sophisticated enough to know about shortcuts and congestion situations as

well as addresses and routes to other stations.

4.2 Initialization and updating

There are two initialization problems. The first one concerns the network as a whole while the second concerns new nodes entering the network. We will consider the first problem as one of minor interest since the network is intended to expand incrementally, starting from very few nodes. Manual procedures can be used to "bootstrap" these nodes. This philosophy puts a lot of burden upon the solution of the second problem, as we later on want to be able to update fundamental node algorithms present in the node at "bootstrap-time". This should be possible to do after that the network is operational and supports a large number of nodes. Another aspect that needs to be taken into account is that of reinitialization in situations where a large part of the network, or maybe the whole network, has been inoperable. Such cases may occur in conjunction with major power failures or as a result of a deadlock due to improper network programming or network overload.

A possible solution to these problems is obtained by dividing the node software into three categories. The first category contains basic software which is permanently installed at all nodes. This software can be simple but needs to be very general in order to allow implementation of various algorithms for routing and channel access. The second category of software contains those programs which control the network and which define its basic properties. These programs describe the actual access method and also include one ore more standardized routing algorithms. Programs in this class are distributed centrally using the network itself. Finally, in the third category, we have user-defined programs which control the individual handling of packets according to the users specification. Programs in the third category can be generated and transmitted over the network by any user.

4.3 Address and location specifications

The routing algorithms in [1] assume that the coordinates of the receiving node are known. This may not always be the case as we want to incorporate mobile stations into the network. Alternative address systems must therefore be available, which circumvent the problem of having to specify exact locations. A more "loose" version of the coordinate system would be to use something similar to the postal codes. The network is divided into groups which are further divided into subgroups. Each group has a name which is unique within each level only. The coordinate system is a special case, in which the groups consist of squares of various sizes.More generally though, the groups may coincide with country borders,

city limits etc. This system opens up the possibility of using nation specific addressing schemes but does not solve the problem with mobile nodes.

The mobile node problem is solved if the routing algorithm can find a node using only its identification code. Generally this code does not carry any geographical information at all. Principally, we can see two different approaches to solve this routing problem. The first approach assumes a random or directed search through the whole network, or a part of it. A directed search is motivated in those cases where we have some a priori information about the location of the node. The second approach is based on the idea that the node continuously updates the network about its location. This information can be sent to nodes which are likely to transmit packets to the mobile node. Alternatively, the location information can be sent to the neighbors of the node's "home"- location or to a suitable station (postal system). Finally, combinations of the two techniques are possible. Either a search can be done for a node that contains information about a particular receiving node or packets can be transmitted into a particular area within which the receiving node is expected to be.

Finally, a note about the coordinate system. A natural choice would be to use the classical longitude/latitude system translated into a suitable binary form. However, spending a few extra bits may return several advantages. Radio amateurs have long used the so called QTH locator which is a system that uses both digits and letters and allows for short form addresses. The QTH locator has been criticized for several reasons. Most seriously is that it is not a worldwide system. In fact, the normal 5 character QTH code has a unique coverage of only about 2% of the globe. Recently a new system, the EA8EX locator, has been proposed. This system offers global coverage. It uses both letters and digits in such a way that short form addresses can be used with varying degree of precision. Typically 6 characters is used to designate a 0.9 * 0.6 km sector (at latitude 45 degrees). Using this system will add about 50% redundancy (2 bytes) to the address field. This drawback is somewhat balanced by the ability to use short form addresses in many situations. Also, as the communication protocol tends to be byte (character) oriented (see part III) the address field can be considered to be "non-encoded clear text" which has a certain intuitive appeal.

CHAPTER 5

5. Network Control

5.1 Traditional control structures

Data networks are usually controlled by information which is transmitted along with the data. In packet switching networks, control bits are commonly found in a particular control field within the transmitted packets. The control field specifies actions to be taken by nodes encountering the packet. These actions may involve the routing strategy, error handling or what particular ACK message should be sent upon arrival of the packet. Similarly, the destination address field can be considered to be part of the control field with the implicit instruction to forward the packet to the specified destination.

Protocols usually follow a hierarchical (or layered structure). By this is meant that one or several packets may be embedded in a larger packet structure in which the previous packets constitutes the data field. Protocol structures on different levels offer the user a number of standardized interfaces to the network. The lowest level is represented by the physical link, while the highest level may correspond to protocols which define the interaction between end-to-end host processes. The main idea behind layered protocols is to "hide" both higher and lower level control information at each level, thereby simplifying the interaction between - and the specification of - the various network protocols. A large standardization effort has been made in order to reach an agreement on what types of levels should be distinguished and how they should be specified. A detailed description of the CCITT X-series of recommendations can be found in [6,chapter 6].

5.2 Suggested control philosophy

From the expected use of the suggested network as an experimental tool, it is evident that the users should have access not only to high level host-to-host facilities, but also to lower level network functions. It should be possible to conduct individual experiments on routing algorithms, control functions, data formats etc. It can easily be conceived that traditional protocol structures do not offer these possibilities. Hierarchical protocol layers can not be defined beforehand. Furthermore, the very idea of hiding certain network features at various levels may be attractive to the naive end user but will only act as unnecessary restrictions in the development of efficient communication procedures. As an example consider a case in which the routing strategy would benefit from knowing what type of data is being transferred (keyboard characters, files, digitized voice, datagrams etc.). Properties such as typical/maximum delay and error probability will be stressed differently depending on what particular type of data is transmitted. This should obviously lead to different choices of routing algorithms. The traditional solution would be to standardize control codes in advance for the various cases. Given a fixed set of hierarchical levels this will lead to a complex set of codes that may still not be sufficient to handle new situations. The major problem is that a hierarchical system does not allow interaction between local and global levels. This fact may turn out to be as major an issue in networking as it is in image processing and other artificial intelligence related fields.

An obvious remedy is simply not to specify a particular solution at all. Instead, all efforts should be put into defining a language in which possible solutions can be expressed. Such a language will define a (logical) machine that can be instructed to perform functions which are specified by the user. The power of this approach is demonstrated by the previously mentioned "envelope" principle. Here, each node would constitute a machine that simply opens all incoming packets, retransmitting those which are intended for other nodes. Figure 6 shows examples of packets intended for nodes C and D as well as a composite packet intended for node B. The composite packet shows a case where subpackets are retransmitted to nodes C and D from where ACK messages are retransmitted to node A. The instruction to perform the "envelope" operation is implicitly given since it is assumed to be the only function of the hypothetical node machine. If we want to express more complex node functions, such as copying messages, then obviously it is necessary to supply explicit instructions within the packet.

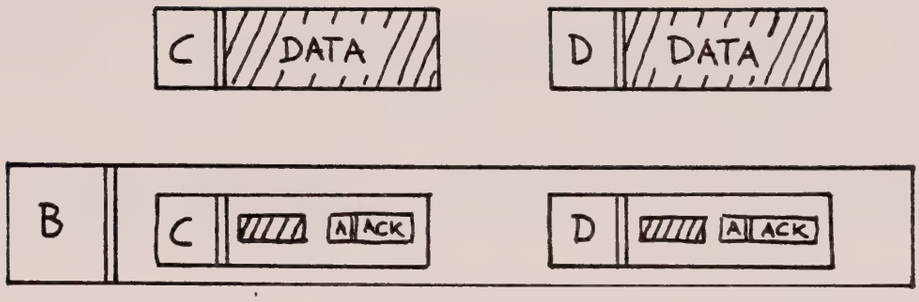


Fig. 6. Example of composite packets

5.3 A Control language

Defining a control language is in principle a simple task. We could use any well defined high level or low level language such as ADA or the assembly language of a popular microprocessor. We would only need to extend the language with application oriented primitives, so that the program can control the transmitter and receiver. There are, however, some basic considerations which have to be taken into account. Program statements will be transmitted along with a packet and be interpreted at a receiving node. This leads to the requirement that the language should be compact. By being compact, we mean that programs written in the language should be short compared to similar implementations in other languages. Furthermore, the language should be incremental in that new language constructs can be defined and used as generally as the basic language primitives. This stems from the requirement that users should be able to program remote nodes to handle their packets in specific ways.

Fortunately there are very few languages which are interpretative, incremental and compact. Out of the four which were studied (LISP,APL,TECO,FORTH) it was found that the ideas behind FORTH matched our application particularly well. This language is a low level, stack oriented language, which was originally developed for controlling Radio telescopes. It has later been adopted in a wider user community due to its extensible structure and ease of implementation.

The next section describes in general terms the process and software structures of a FORTH implementation of the suggested control mechanism. The following section shows some illustrative examples using FORTH-like

notation. A comprehensive description of the FORTH language is found in the Appendix .

5.4 Implementation

A node in the network has to deal with several simultaneous tasks or processes. These can be described as in fig. 7.

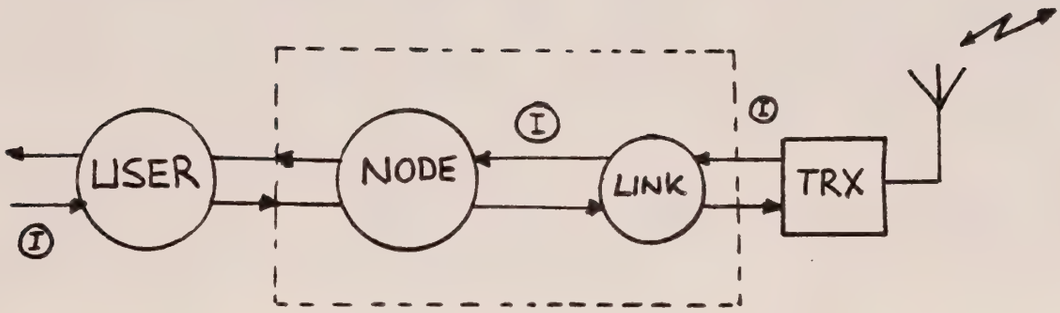


Fig. 7 Processes in the network node

The LINK-process monitors the packet-radio transceiver. Incoming packets are delivered to the NODE-process for interpretation. Outgoing packets from both the USER and NODE processes are handled and transmitted by LINK.

The NODE process handles and interprets incoming packets and decides whether to retransmit them (i.e. return them to LINK), to forward them to USER or to take some other action specified by the packet.

The USER-process handles user interaction and is the users "own" tool for network communication. It packs and unpacks the packets of the user and can deliver them to LINK for transmission. It constitutes the user-interface to the network.

It seems natural to implement the processes as three FORTH-tasks, sharing the same dictionary. The NODE and USER processes interpret their input data stream as FORTH words, i.e. commands which are immediately executed. The input data stream of the NODE-process is the stream of packets arriving from the LINK process. The input stream of the user process are the characters coming from the terminal (computer) line. Several USER-processes can be utilized, one for each terminal line. Communication between processes is achieved trough buffers which hold packets and status variables. The LINK process, for instance, puts all incoming packets in the input-buffer of NODE (after certain preprocessing) for interpretation. In fig 7 all paths marked I indicate interpretation, while the others represent packet or data transfer by means of buffers. As an example one can define a (terminal oriented) USER process that makes the network look like any FORTH-system with interpretation of terminal input line by line.

Since packets are interpreted as commands the user can transmit packets containing defining words to other nodes and thus change the contents (available commands) of these nodes. The user may thereby condition the network to react in a specific way when encountering his future packets. Since almost all words (commands) are available to the user, he can create his own service functions anywhere in the network.

How can this be implemented without causing to much trouble? Again FORTH offers a very elegant solution. A common concept in FORTH is the idea of VOCABULARIES, i.e. subdictionaries. In our case a vocabulary structure according to fig. 8 can be utilized.

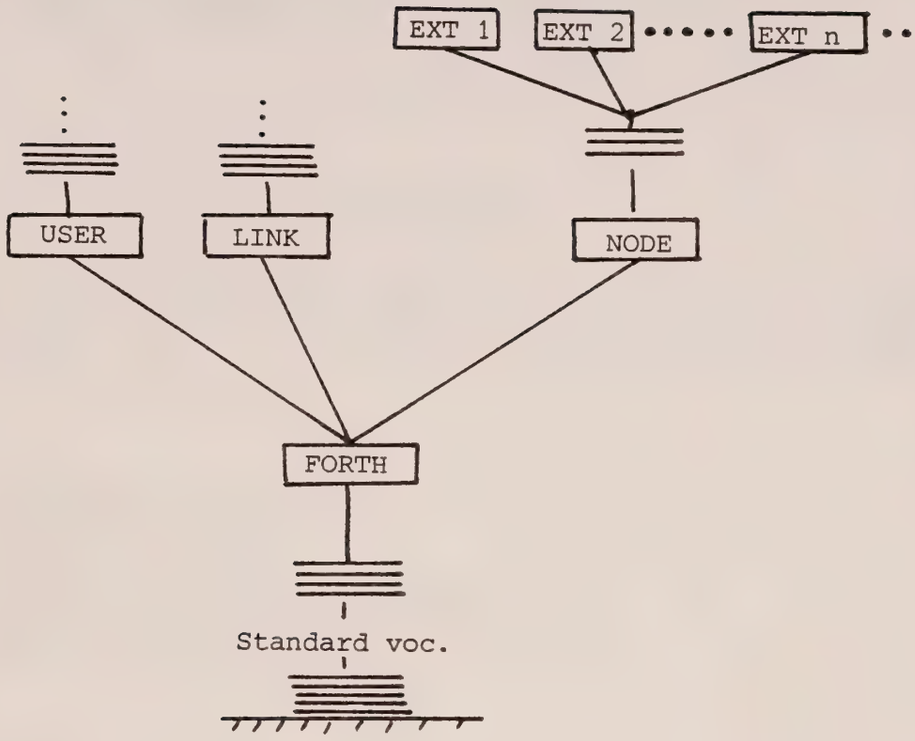


Fig. 8 FORTH dictionary structure

The boxes in the figure symbolize vocabulary names (in FORTH these names are executable FORTH words; by executing them one selects that vocabulary). The different processes in the node select their "own" vocabulary, i.e all the words "downwards" in the diagram are available to them. For instance, the USER-process selects the USER vocabulary, and thus all the words in the USER vocabulary, plus all the words in "standard vocabulary" are available.

External users which utilize the NODE-interpreter are assigned sub-vocabularies denoted EXT1,EXT2,... These are assigned and deassigned to the external network user on a LRU basis (least-recently-used) to prevent overcrowding effects. The available dictionary space is dynamically shared by all these users. To external users the words in their own (EXT) vocabulary, the NODE vocabulary and the standard vocabulary are available. For external users, the vocabulary is switched automatically, by means of the identification of the incoming packet. Thus

commands in the EXT-vocabularies are only valid for packets with the same identification. For safety reasons, the user (and external users), are not allowed to select a vocabulary lower than "their own", and by this gaining control of LINK words or other "restricted" words. Only a privileged user should be able to introduce such changes. To achieve this the FORTH vocabulary word could contain a password mechanism and thus be accessible only to privileged users of the network (see section 5.6). For example all vital node functions, such as radio channel access (LINK-) functions, should be protected against unauthorized manipulation.

5.5 Examples

Besides the obvious network function of transmitting a packet from node A to node B there are a number of functions that will become useful. These include :

Generating ACK packets
Transport services
Counting the number of hops
Generating "programs" in remote nodes
Choosing different routing strategies
Envelope opening function

These functions, as well as numerous others, are implemented by the construction of suitable programs which instruct the logical NODE machine. Consider the previously mentioned example in which node A sends a packet to node B which duplicates the message to nodes C and D. A possible solution is given in fig. 9a. When the packet reaches node B the header information will be stripped of by LINK. The interpreter then finds the message which is treated as a non-separable unit and is pushed on the stack. Next follows the command "DUP" which duplicates the top element of the stack, i.e. the packet. C is an address word and is being pushed on the stack. The command "SEND" forms a packet from the two topmost stack elements and sends the packet by placing it in the transmitter buffer of LINK. The same operation is performed using D as the destination address. A closer look at the messages transmitted to C and D reveals that they contain the word "USER", so that the message is really forwarded to the respective USER process. Furthermore, the word ACK could mean, to produce and send a particular acknowledge packet to the originating node.

In our second example, a program is loaded into node B, in order to perform the previous sequence of operations on a larger number of packets. Figure 9b shows how the facility can be defined into the B node, what the data packets look like, and how the facility can be removed using FORTH-like notation.

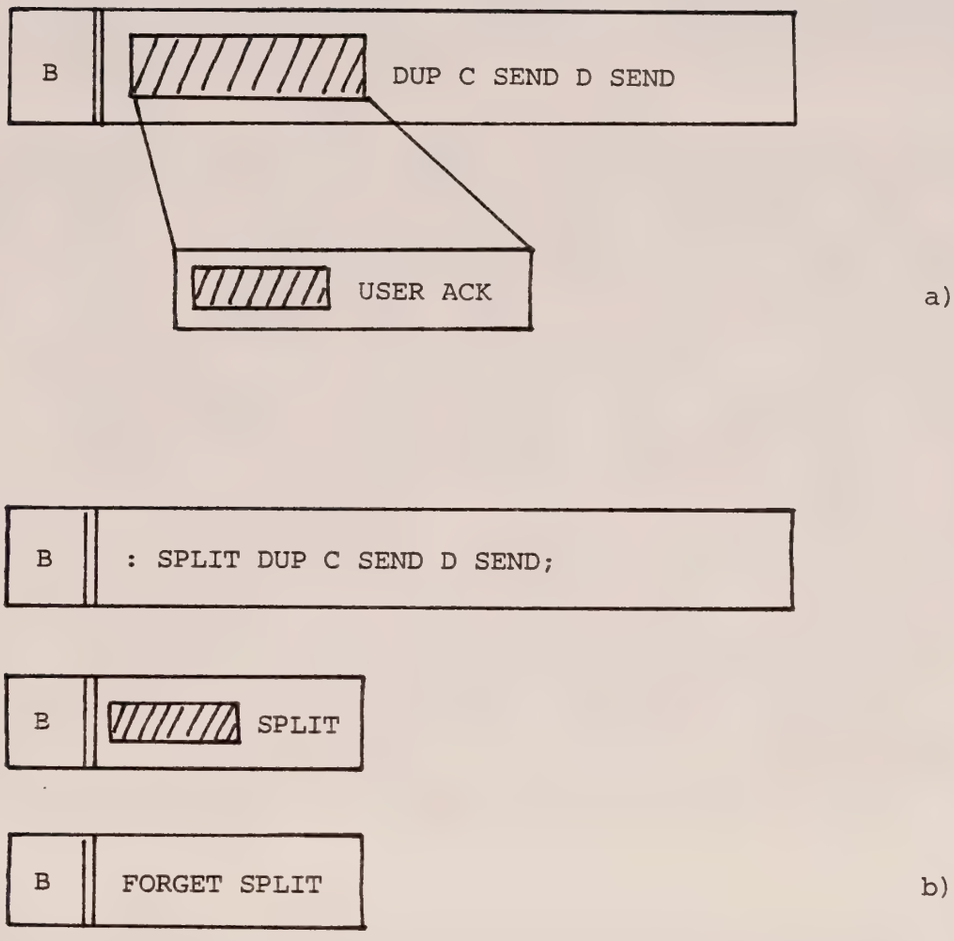


Fig. 9 Example of packets creating new network facilities

If suitable node and packet status variables such as signal strength and last repeating node adress are made available, it is possible to define user specific routing algorithms as well as any other complex node behavior. As a small exercise the reader is invited to define procedures that implement the envelope principle.

5.6 Network integrity

It was previously mentioned that certain parts of the software should only be alterable by privileged users. This means that there must be a procedure implemented in each node that checks the user whenever basic network programs are redefined. Using a secret code is obviously not possible as the complete node software will be available to anyone who wishes to construct a node.

A simple solution to this problem is offered through the use of trap-door functions [7],[8]. These can be described as being encryption algorithms which have different encryption and decryption keys. In our case, the chosen algorithm will be publicly available along with the decryption key. The encryption key, however, is known only to the privileged users. Whenever they want to send a message it is encrypted using the (secret) encryption key. After reception, the receiving node decrypts the message using the public key and checks it for validity (by looking at some predefined redundant data). The privileged user group, which is envisioned here, would consist of people who take an active part in the administration of the network itself. This group would distribute new releases of node programs and act as a coordinator with regard to information exchange and large scale experiments. It should be noticed that nothing prevents the formation of several such groups. In fact, this is highly desirable as it makes possible local (i.e. state or nationwide) control of respective parts of the network. Within such a network domain any access and routing philosophies may be used, independently of what is being used elsewhere. As long as packets entering or leaving such a domain are properly handled, all interior rules and features are defined by the responsible local coordinator group(1). Separation between the domains is guaranteed through the use of different encryption keys.

Further aspects of network integrity and data security based on cryptographic methods can be found in [6,chapter 9] and in [9].

---

(1) In Sweden this will most certainly be a subgroup of the National Radio Amateur Society (SSA).

PART III

IMPLEMENTATION

CHAPTER 6

6. Node Hardware

In an experimental network of the proposed type great care should be exercised when considering the node hardware layout. Future demands are very hard to predict and the hardware should not impose any unneccessary restrictions. There are several considerations that should be taken into account :

1. Flexibility. The hardware should be flexible enough to allow the implementation of future extensions.

2. Laws and Regulations. The frequency band used and maximum allowed channel bandwidth are under governmental control. This imposes constraints on modulation methods and bit-rates etc.

3. Cost. As the nodes are properties of hobbyists, low cost is a crucial point. For a complete node hardware costs should not exceed US$ 1000 .

Furthermore, physical dimensions should be small. The node hardware mainly decomposes into two parts: a microcomputer based control unit and a packet radio transceiver. We will describe these parts and their specifications in that order.

6.1 Control-unit

From fig 7 in Section 5, we see that the hardware has to implement three logical processes of which one has to be "present" in real time. This is the LINK process which monitors the packet radio, i.e the radio channel. The USER process has to be activated at demand, but has no real-time requirements. This holds true also for the NODE process. In the physical implementation the requirements are met by using a two-processor system, one CPU for the LINK process only, and another CPU shared between

the USER and NODE processes by user demand and/or time-sharing. Fig. 10 shows a typical hardware layout with two CPU:s sharing the same bus and memory (and FORTH dictionary). This can easily be implemented with the MC6800 microcomputer chip family. The user interface is asynchronous, V24 standard and uses full duplex at user-selectable speeds up to 38 kbit/s. The packet radio transceiver is interfaced through a synchronous communications interface (SSDA) and a parallel interface for control and status signals. The memory requirements in RAM an ROM are probably not less than 32 Kbytes, preferably more. Several timers are supplied for time slicing and for time dependent link-level tasks.

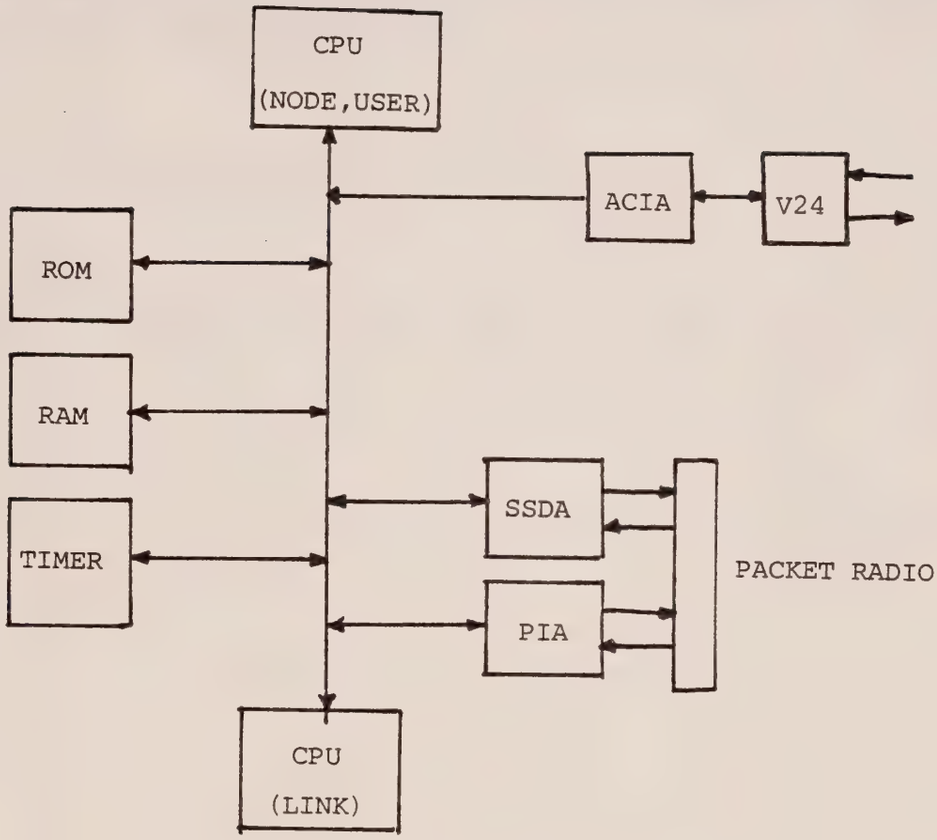


Fig 10 Control-unit (Microprocessor) Hardware

6.2 Packet Radio Transceiver

The LINK process supported by the control-unit interfaces to the UHF radio channel through the packet radio transceiver. Since bit-rates are expected to be moderate (100 kbit/s) slightly modified commercial equipment could be used. The antenna sites for stationary nodes can be selected with some care, which hopefully results in low error rates in the channel. Packet collisions will probably be the dominant source of errors.

Communication on the channel is synchronous and byte oriented. There are several modern LSI interface chips that handle this type of transmission (like the Motorola MC6852 SSDA). The microcomputer system is interfaced with the transceiver section of the node by means of the signals shown in fig. 11 . Bit-level synchronization is obtained in the demodulator which generates a bit-synchronous clock signal. Byte-synchronization is obtained by the SSDA. The receiver also produces a digital signal strength reading S and the transmitter output power can be controlled by the signal "PWR". The number of levels of quantization of these signals has to be considered.

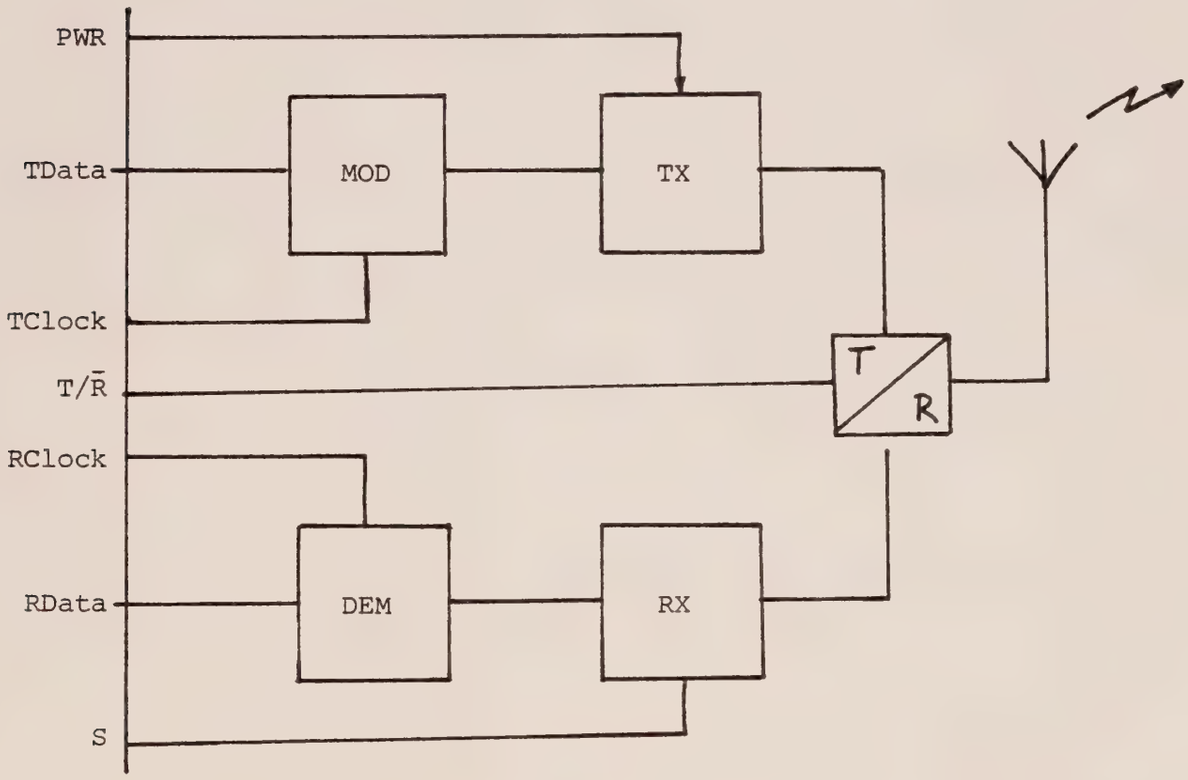


Fig 11 Packet-Radio Transceiver

Crucial problems are the choice of modulation and channel bit-rate since these have great impact on the quality of the transmission path between nodes. Modulation should be bandwidth-conservative and the experimental system should allow a channel bit-rate of at least 100 kbit/s.

CHAPTER 7

7. Preliminary NODE interface specifications

As can be seen from Fig 7, there are two major interfaces which define the NODE interaction: the USER/NODE interface and the LINK/Packet radio interface, here called LINK-interface. The interface between the USER process and the actual user needs no rigorous specification, since the user can specify the USER process freely according to his own demands. The same holds true for the actual configuration of the Packet Radio hardware.

7.1 USER/NODE interface

The USER/NODE interface represents the view the USER process has of the network as a whole. This view consist of two FIFO-buffers, containing packets and status information. In FORTH , these buffers can be described as instances of a new data structure, here called a PACKET-BUFFER. This new structure is described in chapter 8. The interfaces thus consist of two PACKET-BUFFERs :

USB	PACKET-BUFFER for packets destinated to the USER process
TXB	PACKET-BUFFER for packets from the USER process to be transmitted by LINK/NODE.

Besides actual data packets, these FIFO-type buffers also contain status information about the packets, such as addresses, IDs, signal strength and power and finally timing information. Provisions for concurrent access to the buffers are also made.

7.2 The LINK-interface

The LINK-interface is the interface between the packet radio and the LINK process, i.e. the physical packet format. This interface can be described as in fig 12.

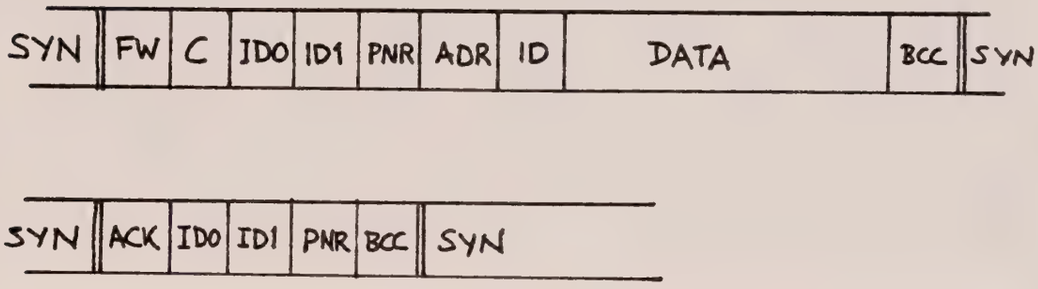


Fig 12 Packet format

SYN	any number of ASCII SYN characters
FW	Format word (interpreted by LINK - process)
C	Byte count (1 byte, length of packet in bytes)
ID1	Immediate destination node ID (1-6 characters)
IDO	Last transmitting node ID (1-6 characters)
PNR	Packet identification number (1 byte)
ADR	Final address (1,2,4,6 characters of EA8EX locator)
ID	Final destination node ID
DATA	Data bytes (interpreted by NODE <128 bytes)
BCC	Block Check bytes (4)
ACK	ASCII ACK character

The format word (FW) controls the interpretation of the remaining packet. Thus the above example can be considered to be one out of several possible structures. Characters are full 7-bit ASCII-characters and the last character in each of the address-fields and FORTH words can have the most significant bit set to act as a delimiter. The BCCs are calculated as the bit-wise modulo-2 sum (XOR) of the bytes in the DATA field in segments of 4 bytes, i.e. the i:th byte can be calculated as

$$(BCC)_i = \sum_{j=1}^{[c/4]} b_{j+i}$$

C	Byte count
[]	Integer part
bj	Byte j
	denotes bit-wise XOR

This is a very simple error detecting parity check code with interleaving. It is well suited for software implementation and has good burst error detecting properties. The code detects all burst up to a length of 32 bits. The capabilities of detecting `pure random` distributed errors are, however, not very good. In case of total carrier lost the probability of correct detection is as small as 2*10E-10. Other blockcodes have been considered, especially the common cyclic codes. They can be optimized either for pure random error detection (CCITT CRC) or for burst errors (FIRE-codes). They are, however, better suited to be implemented in hardware than in software and will not be considered for the moment. The error detecting code used in the system may later on be subjected to change.

CHAPTER 8

8. FORTH extensions.

In this chapter we will suggest some data structures suitable for supporting network node action as described in the previous chapters. There are several fundamental tasks which have to be fulfilled, for example packet handling, public key protection of certain parts of the system dictionary (memory) and reservation of dictionary space (vocabularies) for external users. These problems are discussed in the following in terms of FORTH defining words. We are here merely suggesting words and functions without going into definition and implementational details. FIG-FORTH standard notation [10] for description of words will be used.

8.1 Packet handling

All message exchange in the network is performed by the use of packets. Packets are queued for processing or transmission at several places in every node. It seems natural to introduce a data structure called PACKET-BUFFER, to handle these operations. The PACKET-BUFFER consists of packets and associated status information, forming a linked list implementation of a FIFO stack. Among the status information associated to each data packet we have the addresses and IDs shown in fig 12. There, we also have the time of packet arrival, signal strength and/or transmitter power. As packets are entered into or retrieved from the buffer by concurrent processes, a semaphore, protecting the buffer from simultaneous accesses by two processes, is introduced. The defining word operates like this:

PACKET-BUFFER n ---
(--- addr executing)

Creates a packet buffer for at most n packets with the name cccc in the dictionary. Used in the form n PACKET-BUFFER cccc . When cccc is later executed it returns the address of the semaphore field of the buffer.

In the following we suggest some useful words for packet and packet-buffer handling.

CLBUF addr ---

Empties buffer at address given on stack. This is achieved by linking all packet spaces to the free list.

INPACK addr1 --- addr2

Gets address (addr2) for next available packet space in the buffer indicated by address on stack (addr1)

OUTPACK addr1 --- addr2

Gets address (addr2) of the uppermost packet space in the buffer at addr1.

FREEPACK addr ---

Remove the uppermost (least recent entered) from the buffer at addr (Link to free list)

REMPACK addr ---

Remove packet at packet space address addr. (Link it to free list)

WAIT addr ---

Decrement buffer semaphore for packet buffer at addr. (Disables interrupt while executing)

SIGNAL addr ---

Increment buffer semaphore for packet buffer at addr. (Disables interrupt while incrementing)

LID addr1 --- addr2

Get address (addr2) of the Last/Immediate destination node ID for the packet space at addr1.

FADR addr1 --- addr2

Gets address (addr2) of the final destination for the packet space at addr1.

FID addr1 --- addr2

Gets the address (addr2) of the final destination

node address (locator) for the packet packet space at addr1.

PTIME addr1 --- addr2

Gets address (addr2) of the packet arrival time for the packet space at addr1

S/P addr1 --- addr2

Gets address (addr2) of the signal strength or power for the packet space at addr1

PACKET addr1 --- addr2

Gets address (addr2) of actual data packet in packet space at addr1.

8.2 Protected dictionaries and vocabularies

As the network is indeed to be a public one, with the nodes being the property of the users, some rather elaborate protection measures have to be taken to secure the most vital functions of the network. These are functions like LINK- and Packet Radio control and some low level routing algorithms which should be available at any time. The simplest way to protect these facilities is to supply them in ROM storage, as is done with the "boot-up" dictionary. This, however, will not work in those cases, where changes are to be made at runtime by privileged users. In those cases the algorithms can be stored in a vocabulary protected by a trap-door function as mentioned in chapter 5.6. In fig 8 for example, the FORTH and LINK vocabularies could be protected vocabularies. The defining word can be of the following form :

PROVOC n1 n2 n3 n4 --- (compiling)
n1 n2 n3 n4 --- (run-time)

Creates a new protected vocabulary in dictionary with the key n1,n2,n3,n4. Used as PROVOC cccc, which creates a vocabulary cccc. At runtime when cccc is executed the vocabulary is selected if the correct key n1-n4 is on the stack. Otherwise an error-message is issued.

8.3 External user vocabularies

It was earlier mentioned that external users share available dictionary space on a LRU basis. When the allowed space for external users becomes full, the least recently used external vocabulary is deleted to make room for a new user. Two defining words EXTDICT and LRUVOC are used to define this data structure.

EXTDICT n --- (compiling)
  --- addr (runtime)

Creates a LRU - dictionary of n bytes.
Used as EXTDICT cccc. At runtime cccc
returns the base address of the dictionary

LRUVOC addr --- (compiling)
addr --- (runtime)

Conditional compiler. Used as LRUVOC cccc
If cccc does not exists a vocabulary is
created, otherwise the vocabulary is just
selected.

CHAPTER 9

9. Summary and conclusions

As a complement to [1], we have outlined the specifications for a potentially global wireless data communication system for radio- and computer amateurs. The system is a packet switching network using UHF radio. In the network every participant has his own network node and message transfer is made through distributed processing in the nodes with no hierachical control. In the report we touched upon several issues regarding the implementation of such a network. First we studied inter-node communication aspects. A simple multiple-access scheme of the CSMA class was introduced together with an approximate analysis giving some rough upper and lower bounds on throughput- and delay-performance. The problem of packet routing through the network was discussed. It is relevant to conclude that the simple algorithms described in [1] should be used primarily to "bootstrap" more sophisticated strategies under software control. The problem of network control gave rise to the introduction of a control language. A survey of some existing computer languages indicated that FORTH was suitable for network control purposes. With this tool the user is able to condition (program) his own and other nodes at all levels to meet his demands. This can be achieved by allowing the user to run programs (processes) in remote nodes.

Finally, we outlined the hardware and software implementation of a network node. The hardware consist of an UHF packet radio and a microcomputer system, while the associated software is a multi-task FORTH system with special purpose vocabularies. The hobbyist should be able to build his own hardware, which has to be low-cost and reliable. The software should be standardized and publicly distributed.

CHAPTER 10

10. References

[1] Persson,I.E., Forchheimer,R. : "Design considerations of a distributed packet radio network using the radio amateur band". Internal report LiTH-ISY-I-408.

[2] Binder,R.,Abramson,F. et al. : "ALOHA packet broadcasting-A retrospect" NCC, 1975.

[3] Kahn,R.E. : "Advances in packet radio technology", Proc. IEEE,nov 1978.

[4] Kleinrock,L.,Tobagi,F.A.. : "Packet Switching in radio channels", part I+II, IEEE Trans. Comm. Dec 1975.

[5] Kleinrock,L.,Yemini,Y. : "An optimal adaptive scheme for multiple-access broadcasting", Conf Rec ICC'78, Toronto,June 78.

[6] Barber,D.L.A.,Davies,D.W. : "Computer Networks and their protocols", Wiley Interscience, 1979.

[7] Diffie,W Hellman,M.E. : "New directions in cryptography, IEEE Trans. Inf. Theory, IT-22,644, Nov 1976.

[8] Davies,D.W. et al : "Evaluation of Public Key Crypto Systems", Information Privacy, Vol II nr 4, July 80.

[9] Blom,R. et al : "Encryption Methods in Data Networks", Ericsson Technics, No.2, 1978.

[10] FIG-Forth glossary, Forth Interest Group.

FORTH - an introduction

Computer languages are usually designed for a specific purpose. When implementing an algorithm, these languages impose a rigid framework upon the programmer. The basic concept in FORTH is, however, to provide the programmer with a minimum set of constructs that will be used to build any new construct, entities, data structures etc.

FORTH programs does not look like BASIC or Pascal-programs, but are built up by system- and user-defined functions or procedures which are called words. Each word describes an object or an action. The user defines his own words as combinations of other words, which in their turn call other words and so on. The program development is interactive. Words can be defined and tested directly and give immediate feedback. The interactivity is used for stepwise refinement.

The elements of FORTH

The stack

To pass parameters and returning results we use a stack, which we operate via postfix (RPN) notation. This has several advantages:

Parameters are easily passed between different modules (words)

Local variables are kept on the stack and need not to be named

Module testing is very simple. The user can easily enter suitable parameters on the stack an execute the module and then examine the stack for results.

The dictionary

All FORTH constructs, as they are defined are stored in a linked list of words - the dictionary. The user has the possibility to create his own sub-dictionaries, vocabularies to suit certain applications.

The interpreters

The interactive FORTH system contains two interpreters, the inner interpreter, also called the address interpreter and the outer interpreter, called the text interpreter. The latter one, interprets text comming from the terminal input as words and executes them. A special type of words are defining words or the compilers. These words can create new entries in the dictionary. Such a word is ´:´. An example

```
: SQR DUP * ;
```

We here define the word ´SQR´ as the sequence of the words ´DUP´ and ´*´. The word ´;´ is used to conclude the definition. At ´compile-time´, i.e. when we enter the text line above, a dictionary entry with the name ´SQR´ is formed. This entry contains besides the name ´SQR´, also the execution addresses of the words ´DUP´ and ´*´. When we later execute ´SQR´ these addresses are the interpreted by the inner interpreter. This interpreter does not need to bother about searching for the words ´DUP´ and ´*´ in the dictionary, this has already been done by the text interpreter and the word ´:´. This form of ´semi-compilation´ offers a great advantage with regard to execution speed.

Continuing the example above we can now follow the execution of the module ´SQR´ by hand. We enter

```
4 SQR .
```

When ´4´ is encountered it is interpreted as a number and pushed on the stack (assuming there is no defined word ´4´). Then we execute ´SQR´, which contains the sequence ´DUP´ ´*´ . The word ´DUP´ is a predefined one and duplicates the top element of the stack. The stack now has following content

```
 4   ( top-element )
 4
---
---
```

´*´ multiplies the two top elements and leaves the single number 16 at the top of the stack. This number is finally printed on the terminal by the print word ´.´ .

Further examples of FORTH programs can be found in the July-19 1980 issue of Electronic design: ´FORTH - an extensible path to effective programs´, and in the august 1980 issue of BYTE magazine, entirely devoted to FORTH. The proposed standard for the predefined set of word used by the FORTH Interest Group can be found in [10].